CORDIC based SSIM Computation in FPGA

Sanjit Dhali

Department of Electrical Engineering

National Institute of Technology, Rourkela,

Rourkela-769008, Odisha, India.

May 2015

CORDIC based SSIM Computation in FPGA

Dissertation

submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

by

Sanjit Dhali

Roll No: 213EE1282

under the guidance of

Dr. Supratim Gupta

Department of Electrical Engineering

National Institute of Technology, Rourkela

Rourkela-769008, Odisha, India.

May 2015

Dedicated to

My Parents

National Institute of Technology, Rourkela

CERTIFICATE

This is to certify that the thesis entitled, **" CORDIC based SSIM Computation in FPGA"** submitted by **Sanjit Dhali** in partial fulfillment of the requirements for the award of Master of Technology Degree in **Electrical Engineering** with specialization in **Electronic Systems and Communication** during 2014-2015 at the National Institute of Technology, Rourkela is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Date: .....

Dr. Supratim Gupta
Dept. of Electrical Engineering
National Institute of Technology
Rourkela-769008
Odisha, India

"Education is the ability to listen to almost anything without losing your temper or your self-confidence."

$\sim$ Robert Frost

Abstract

Images may be distorted by imaging system or by transfer it from one system to another or to process for different application. So, it is important to calculate the quality of the image before presenting to the viewer. This distortion measured by some traditional parameters like PSNR, SSIM and RMSE. But, SSIM is well established method, provides nearly same result as human visual system. The distortion in the image has been calculated using SSIM in this work. The compressed SSIM has been measured in this work in frequency domain in Discrete Cosine Transform(DCT) dictionary. Images are 2D data matrix, so this work aims to calculate 2D DCT using 1D DCT. 1D DCT is calculated by well established CORDIC algorithm. The CORDIC algorithm is sequential in nature.But can also be implemented parallel using FPGA, as different processes in FPGA produces output simultaneously. The output of VHDL core is compared with MATLAB inbuilt SSIM function's output.

Acknowledgments

I express my sincere gratitude towards my guide **Dr. Supratim Gupta** for his constant help, encouragement and inspiration throughout the project work. Without his invaluable guidance, this work would never have been a successful one.

I am also very obliged to Prof. Anup Kumar Panda, HOD, Department of Electronics and Communication Engineering for creating an environment of study and research. I am also thankful to Prof. P.K.Sahoo, Prof. Susmita das, Prof. Dipti Patra, Prof. K.R.Subhashini for helping me how to learn. They have been great sources of inspiration. I would like to thank all faculty members and staff of the EE Department for their sympathetic cooperation.

It would have been impossible to walk through this course without bless and help of my family members. I would also like to thank the members of the *Embedded System and Real Time Lab* at NIT Rourkela, namely Susanth Panigrahi, Lazarus, Suresh Gujar, Dipshika and Amit for their valuable suggestions and helpful discussions.

Sanjit Dhali

National Institute of Technology Rourkela

May 26, 2015

Contents

List of Figures

Chapter 1

INTRODUCTION

Most of the image processing techniques are consecutive in nature. These techniques are generally validated using high-level languages like C, C++, MATLAB and Java, etc. which are more accurate. These platforms are used when response time is not a matter of concern. But it will not suit where response time has great importance.

For real-time image processing applications, some fast and time-bound system is required. Micro-controller or DSP chip can be used to execute for the real-time system due to its advantages like low power consumption, portable and economical in cost.

The main problem with micro-controller is its instruction execution that is sequential. So the problem being the number of clock cycles required is more. The speed of execution can be overcome with advanced processors. These processes use pipeline, and super-scalar architectures. These propelled central processing unit consolidate parallel operation at the instruction level. However, the overall execution of the technique will be successive in nature. In this manner, a micro-controller based framework can't successfully use the characteristic parallelism included in the majority of the image processing techniques. This will impact on the general processing rate. Hence, such systems are not suitable for time primary applications.

The advantages in the FPGA can get over the problem with sequential execution and efficient performance. Field Programmable Gate Arrays (FPGA) give

a facility for parallel execution. In an FPGA-based design with high frame rate, the different process accomplishes the sequences of an techniques in parallel and thus supplies a quick response. In FPGA-based hardware the overall operation takes less number of clock cycles, which leads to less power consumption, when compared with micro-controller and DSP-processor based systems.

1.1 Motivation of the project

1.1.1 Parallel computation of SSIM in DCT domain

Structural Similarity Index Measurement (SSIM) is the measure of similarity between two images of an object. SSIM gives a better similarity measure compared to traditional methods like MSE and PSNR. So our concern is to calculate SSIM between two images of the same object.

The SSIM is mainly computed in the spatial domain. It can be evaluated in the frequency domain as compress sampling gives the samples in the frequency domain. So SSIM calculation in the frequency domain can be done to evaluate the perceptual similarity of compressed sample images.

The primary motivation of our project is to speed up the image processing technique, which can be done by FPGA platform as discussed in the introduction. This FPGA platform can give parallel architecture for SSIM calculation in DCT domain or frequency domain. Some of the advantages of FPGA which are motivated to the designer are described below.

Advantages of FPGA

Performance:

Parallelism in the FPGAs, it substituted the DSP chip and micro-controller in some specific cases.

Time to market:

The flexibility of FPGA and rapid prototyping capabilities offers increased

time-to-market concerns. It provides a platform to test an idea or concept to verify it in hardware with less cost compared with ASCI design [12][13].

Cost:

The non-recurring engineering (NRE) cost of custom ASIC outline far surpasses than FPGA-based equipment arrangements.

Reliability:

Test bench and synthesis report of gives visualization to FPGA hardware that is genuinely a "hard" usage of system recreation. If there should be an occurrence of processor-based frameworks when include a few layers of deliberation to help calendar assignments and offer assets among different procedures.

Long-term maintenance:

FPGA chips can be field-up-gradable and don't oblige the time and cost as in ASIC.

1.1.2 Implementation of CORDIC algorithm to calculate DCT coefficient

DCT computation is the measure of frequency, strength in an image. This DCT can be calculated using the cosine function that depends on the spatial strength of the image. This cosine value shows the strength of the frequency component. There are many different technique to calculate the cosine value, here we have chosen CORDIC algorithm in this work for its inherent advanced quality. The CORDIC algorithm can compute cosine, sine, logarithmic, etc. function with the help of simple adder and shifter that is very simple to design and low cost effective. This feature made it famous among the core designer.

1.2 Objective and scope of the project

The objective of the work is to design a parallel architecture for SSIM estimation in DCT domain that categorized the project goal as follows

- Validation of CORDIC in MATLAB for calculation of Cosine function
- Design hardware for CORDIC algorithm to calculate Cosine function
- 1D DCT calculation using CORDIC algorithm
- 2D DCT calculation using 1D DCT Coefficients
- SSIM calculation In DCT domain from DCT coefficient

Using this design a real-time system can be made, that can produce the output for estimation of the similarity measure index of the image.

1.3 Literature Review

According to the objective of our work our literature review can be categorized hardware based system design, hardware design for image processing and full reference quality index measurement for image quality assessment.

1.3.1 Hardware based systems

An embedded system is a hardware computer system that is dedicated to specific functions to serve in a larger mechanical or electrical system when response time is the primary concern. Embedded systems are very general purpose systems due to its qualities like low power consumption and economical low cost. The micro-controller and DSP, embedded systems has the disadvantage of instruction execution, it executes the instruction sequentially [9][16]. The sequential execution starts which fetches the instruction appended by decoding an execution. After fetching it starts fetching the next code while decoding the previous code. So it

is inherently a sequential system that will take more clock cycle to execute an algorithm. So our primary goal is to develop a hardware that executes an algorithm with less number of clock cycles. This facility is available in FPGA platform.

A field-programmable gate array (FPGA) is an integrated circuit designed which can be arranged by a client or a designer subsequent to assembling, consequently "field-programmable", which is the most favorable advantage of FPGA over micro-controller. We can have the processes done according to the written HDL code "in parallel" which means simultaneously. The ability of FPGA parallel processing is one of the most important features that separate FPGA from the processors and made it superior in many areas. This facility is adopted in our work.

1.3.2 Hardware base image processing

The image processing techniques are validated using some high-level languages that will increase the bugged, and the algorithms are also sequential. Embedded micro-controller can be a better choice for design and economic cost system. From the above discussion, we have concluded that, for real-time operation the embedded system like micro-controller and DSP, not a right choice for real-time image processing system [3][7].

FPGA platform is giving a better advantage to implementing the sequential image processing algorithms in the parallel platform. The FPGA hardware works as shown below.

In FPGA, the processes are concurrent means the produces output simultaneously. So the algorithms can process in parallel. Due to this advantage in FPGA we chose FPGA as the hardware platform.

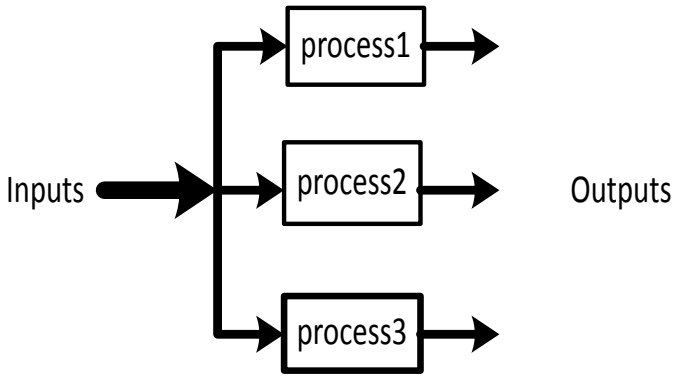

Figure 1.1: Working of VHDL processes.

1.3.3 Full Reference Image Quality Index (SSIM, PSNR)

The image quality is a characteristic measure of an image which can relate to the perfect image. The degradation in the image due to image processing or imaging system can be quantified in subjective quality measure or objective quality measure.

The subjective quality index considers the human visual system (HVS) characteristics to measure the distortion in the image. The example of subjective assessment technique is the mean opinion score (MOS). The MOS is not accurate enough when the images have minor changes or distortions. This method is time-consuming, as it is done by the human. This method is not preferred in the real-time based systems.

Objective quality measures the distortion in the image by mathematical defined model. Target image quality measurements can be arranged by accessibility of an original or distortion-free image. The distorted picture is compared with the original image with evaluating the measure the noise strength in it. This assessment is categorized into three types, and these are described below.

Full-reference image quality measure the quality of the distorted image with available original distortion-free image. That means the user has an original non-

distorted image of the distorted image. When there is no information available about the original image, then no reference or "blind" assessment of the image is preferred. This situation occurs in most of the practical scenarios [8].

Sometimes we have some information about feature, corner, etc. about the original or referred image that we can use as a reference to evaluate the distorted or noisy image. Then this type of assessment is called reduced-reference quality assessment [1].

This work concentrates on full-reference picture quality assessment. There are different models to measure quality, for example, mean square error (MSE), signal to noise ratio (SNR), pick signal to noise ratio (PSNR), mean absolute error (MAE), and root mean square error (RMSE). But from these complicated objective metrics has not shown any clear advantages over simple metrics like PSNR and RMSE. Still objective quality measure is popular due to of its advantages. First, it is easy to calculate and less computational complexity. Secondly, it does not depend on viewing conditions and the observer. Zhou Wang et al. has proposed a structural similarity measure index (SSIM) which gives a comparatively better result in some conditions.

Later on the controversy between PSNR and has shown their relation related with image quality measure. It has been described by Hore et al [2].

1.3.4 SSIM in DCT domain calculation

It seen from sampling theorem, that the representation of the sparse signal in some basis of the dictionary is possible with less number of coefficients. DCT is a commonly used sparse representation dictionary for image signal in the frequency domain. So, it can help to reduce the memory allocation for storing the signal for further use as in JPEG. Also, it reduces the bandwidth of channel. So it will be very convenient that if we can find the information about the image in DCT

domain, then we can analyse that this image is acceptable for display or not. DCT is a unitary transform, and it obey the Parseval theorem. Using this property and SSIM spatial equation, the relation between spatial components in terms of DCT coefficients can be found out [sankar sir thesis]. To speed up the operation, it is required to do some parallel architecture to find SSIM in DCT domain. So the assessment of distortion of the image in DCT domain is possible. So we can calculate the SSIM in DCT domain.

1.4 Thesis organization

- **Chapter 1:** *Introduction-* Measurement of quality of image is described in this chapter. Briefly, SSIM is described for image assessment. Advantages of hardware design in FPGA also described.
- **Chapter 2:** *The CORDIC Algorithm-* Basics of CORDIC algorithm and its application to calculate sine and cosine function is described in this section. Hardware of CORDIC also designed.
- **Chapter 3:** *DCT computation-* DCT is described. Hardware of DCT is designed and compared with MATLAB inbuilt function's output.
- **Chapter 4:** *SSIM computation-* SSIM is described both in spatial and frequency domain. Hardware of SSIm in frequency domain also designed among with it's parameter. Result of designed hardware is compared with MATLAB inbuilt function's output.
- **Chapter 5:** *Test bench for evaluation of SSIM through hardware-* In this chapter briefly test bench is described and the result of MATLAB spatial domain approach and VHDL frequency domain approach is plotted.
- **Chapter 4:** *Conclusion and Future Work-* In this chapter conclusion of this thesis and the research directions for future work have been compiled.

Chapter 2

The CORDIC Algorithm

The CORDIC algorithm stands for CO-ordinate Rotation DIgital Computer. Jack E. Volder invented it in 1959. In 1971 generalized the CORDIC algorithm to compute hyperbolic, circular and linear coordinate system by John Walther. In CORDIC algorithm, the simple ancient principle and two-dimensional geometry is adopted [6]. The main reason for the adaptation of CORDIC algorithm by hardware designer is due to its simplicity and low-cost hardware. The rotation of vectors through a fixed angle, to know and known angle has an extensive application in graphics, robotics and animations. With simple shifting and addition operation, it can compute trigonometric function, square root, multiplication, and division. CORDIC algorithm may not be the quickest system to perform these operations, still it is appealing due its chasteness [4].

2.1 Basics of CORDIC

Let us consider an example to understand the implementation of CORDIC algorithm. In this example we will discuss about the rotation mode vector.

The rotation of a $2-D$ vector with an angle θ, to obtain a final rotated vector. Here the initial vector and rotated final vector are $P_0 = \begin{bmatrix} x_0 & y_0 \end{bmatrix}$ and $\begin{bmatrix} P_n = x_n & y_n \end{bmatrix}$ respectively. This operation can be carried out by the matrix simple matrix product $P_n = RP_0$, where R is rotation matrix that helps to rotate

Figure 2.1: A picture of a gull.

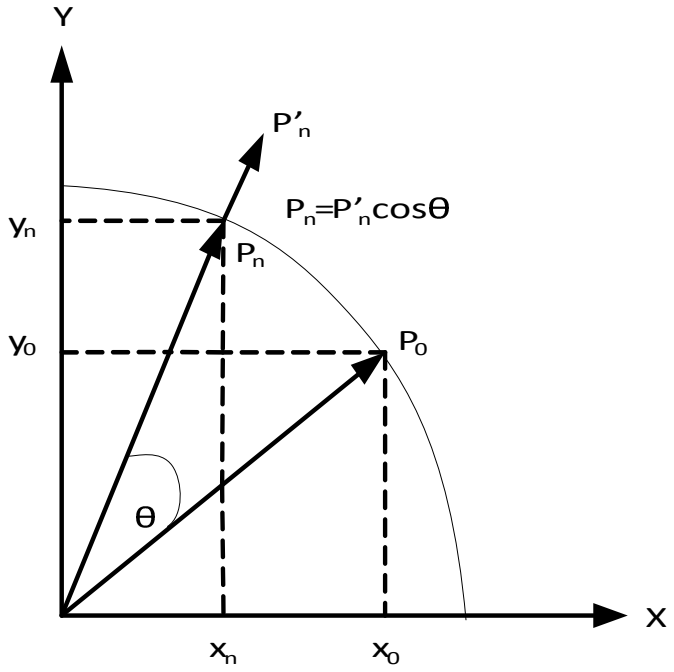

Figure 2.2: MATLAB input to VHDL core

the initial vector to final rotated vector. Where R can be written as

$$R = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \tag{2.1}$$

With simple mathematical deduction the above equation can be re-written as:

$$R = \left[(1 + tan^2\theta)^{1/2}\right] \times \begin{bmatrix} 1 & -tan\theta \\ tan\theta & 1 \end{bmatrix} \tag{2.2}$$

$$R = K \times R_c \tag{2.3}$$

where

$$K = \left[(1 + tan^2\theta)^{1/2}\right] \tag{2.4}$$

and

$$R_c = \begin{bmatrix} 1 & -tan\theta \\ tan\theta & 1 \end{bmatrix} \tag{2.5}$$

Here, K is called as scalar-factor and the matrix R_c as pseudo-rotation matrix. The pseudo-rotation matrix produces the pseudo-rotated vector that can be

converted into the final rotated vector by multiplying with a factor $K = cos\theta$. The above rotation of vector can be implemented in hardware using the following procedure for CORDIC (i) disintegrate the rotations into a succession of elementary rotations done by specified angles that could be actualized with the least equipment expense; and (ii) to abstain from scaling, that may include arithmetic operation, for example, square root and division. The second thought is in view of the reality the scale-factor holds just the magnitude data, however no entropy about the angle of rotation.

2.1.1 Iterative Decomposition of Angle of Rotation:

The CORDIC algorithm executes the rotation repeatedly by separating the angle of rotation into an arrangement of little predefined angles, $\alpha_i = arctan(2^{-i})$, so that $tan\alpha_i = 2^{-i}$ could be executed in hardware by shifting through i bit locations. As opposed to executing the rotate straightforwardly through a angle θ, CORDIC executes it by a specific number of micro-rotations with angle α_i, where

$$\theta = \sum_{i=0}^{n-1} \sigma_i \alpha_i \qquad \text{where} \quad \sigma_i = \pm 1. \tag{2.6}$$

The above equation satisfy the CORDIC convergence theorem which states that $_i - \sum_{j=i+1}^{n-1} \alpha_j < \alpha_{n-1}, \forall i, i = 0, 1, 2, ..., n-2$,but from the above equations we can see that the convergence range is limited to $-1.743291.74329$. These equations shows that $= \sum_{i=0}^{\infty} {}_i = 1.74329.....$ Mathematical derivations shows that rotation is limited in first and fourth quadrant. With non-restoring decomposition

$$w_0 = 0 \quad \text{and} \quad w_{i+1} = w_i - \sigma_i \alpha_i$$
$$\text{where} \quad \sigma_i = \begin{cases} \sigma_i = 1, & \text{if } w_i \geq 0 \\ \sigma_i = -1, & \text{otherwise.} \end{cases} \tag{2.7}$$

Where the rotation matrix for the ith iteration standing for the known small angle α_i is given by

$$R(i) = K_i \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \tag{2.8}$$

Where $K_i = 1/\sqrt{1+2^{-2i}}$ called scalar factor. Then the pseudo-rotation matrix can be written as:

$$R_c(i) = \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \tag{2.9}$$

So, from the above equation of pseudo-rotation matrix, it can be observed that the *ith* iteration of the algorithm rotated with an scalar factor $K_i = 1/\sqrt{1+2^{-2i}}$, which is independent of the value of α_i (direction of micro-rotation) used in the angle decomposition.

2.1.2 Avoidance of Scaling in Instantaneous iteration

The other modification carried out by the Volder's algorithm is to debar the scalar-factor $K_i = 1/\sqrt{1+2^{-2i}}$. By taking out the scaling factor the iterative micro-rotations provide a pseudo-rotated vector $P'_n = R_c P_0$ in place of wanted rotated vector $P_n = K R_c P_0$, where the scale-factor K is given by

$$K = \prod_{i=0}^{n} K_i = \prod_{i=0}^{n} 1/\sqrt{1+2^{-2i}} \tag{2.10}$$

From eq [2.10], it can be observed that the iterative value of the scale-factor of micro-rotations independent of its iterated direction of micro-rotations and diminishes gradually. Thus, the final value of scale-factor K converges to 1.6467605. Therefore, to avoid extra multiplication during each micro-rotation, the value of final output of pseudo-rotation could be multiplied by K. So, the basic CORDIC micro-rotations are found by using the pseudo-micro-rotation of the vector to have, $P'_{i+1} = R_c(i) P_i$, in concert with the non-restoring decomposition of the predefined angles α_i, as follows:

$$\begin{aligned} x_{i+1} &= x_i - \sigma_i 2^{-i} y_i \\ y_{i+1} &= y_i + \sigma_i 2^{-i} x_i \\ w_{i+1} &= w_i - \sigma_i \alpha_i \end{aligned} \tag{2.11}$$

According to the mode of operation CORDIC algorithm can be divided as follows.

2.1.3 Types of CORDIC

The CORDIC iteration could be operated in two types of mode. These two mode basically differ by their direction of rotation. These are described below

Rotation Mode

In this mode, the initial vector in rotated with already given θ. The direction of the rotation of the vector is given by the sign of the instantaneous anglew_i. Where $\sigma_i = 1$, if $w_i \geq 0$ and $\sigma_i = -1$ the iteration has to be continue until satisfying accuracy.

Vectoring Mode

In this mode of operation the direction of the pseudo-rotation is depends on the sign of the y_i. Where $\sigma_i = -1$, if $w_i \geq 0$ and $\sigma_i = 1$ the iteration has to be continue until satisfying accuracy.

After every succession, the quantity of shifts is increased by a couple of barrel-shifters. To get an n-bit output accuracy, $(n + 1)$ CORDIC cycles are required. It can be observed that CORDIC could be actualized by a straightforward choice from the original proposed serial architecture or in completely parallel CORDIC architecture, where no barrel-shifters are included.

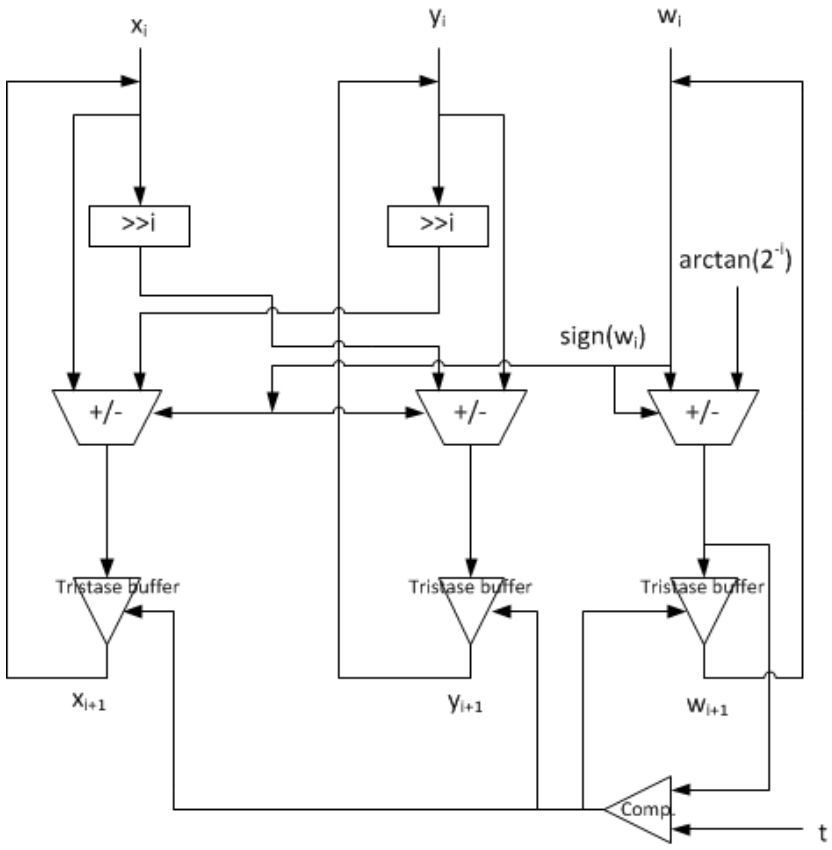

Figure 2.3: Hardware implementation of CORDIC iteration rotation mode.

The principle step has been taken to beat the issue of the restricted convergence range and, then to stretch the CORDIC rotations to the complete scope of $\pm\pi$, an additional cycle is obliged to be performed. This new iteration is indicated in next mathematical statement which is needed as an initial rotation through $\pm\pi/2$. The initial rotation shown in eq[2.12].

$$\begin{aligned} x_0 &= -\sigma_{-1} \times y_{-1} \\ y_0 &= \sigma_{-1} \times x_{-1} \\ w_0 &= w_{-1} - \sigma_{-1}\alpha_{-1} \\ \text{where,} \quad \alpha_{-1} &= \pi/2. \end{aligned} \tag{2.12}$$

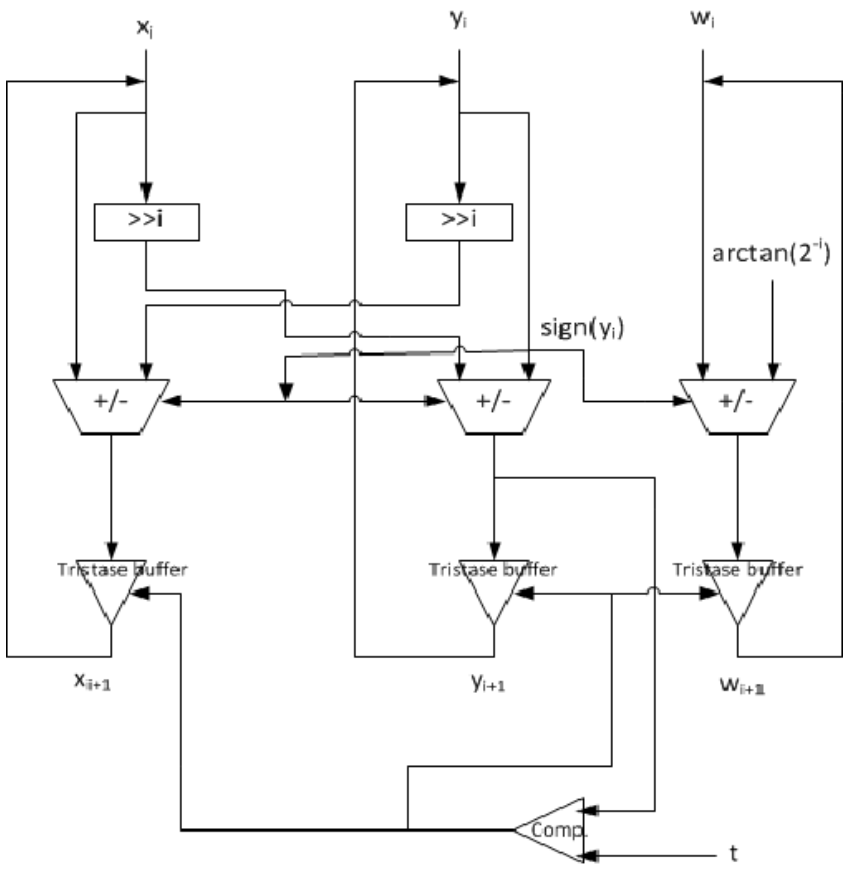

Figure 2.4: Hardware implementation of CORDIC iteration vectoring mode.

2.2 Implementation of CORDIC in MATLAB

The CORDIC algorithm is evaluated using MATLAB. A graphical user interface (GUI) has been develop to verify the accuracy of CORDIC algorithm. In this GUI the inputs are two co-ordinate of vector originated from origin of the axes, number of micro-rotation, and angle of rotation to produce the final rotated vector.

In GUI two operations are available

1. To plot the original vector.

2. To plot the original vector.

The GUI output for evaluation of CORDIC algorithm is shown in fig[2.5].

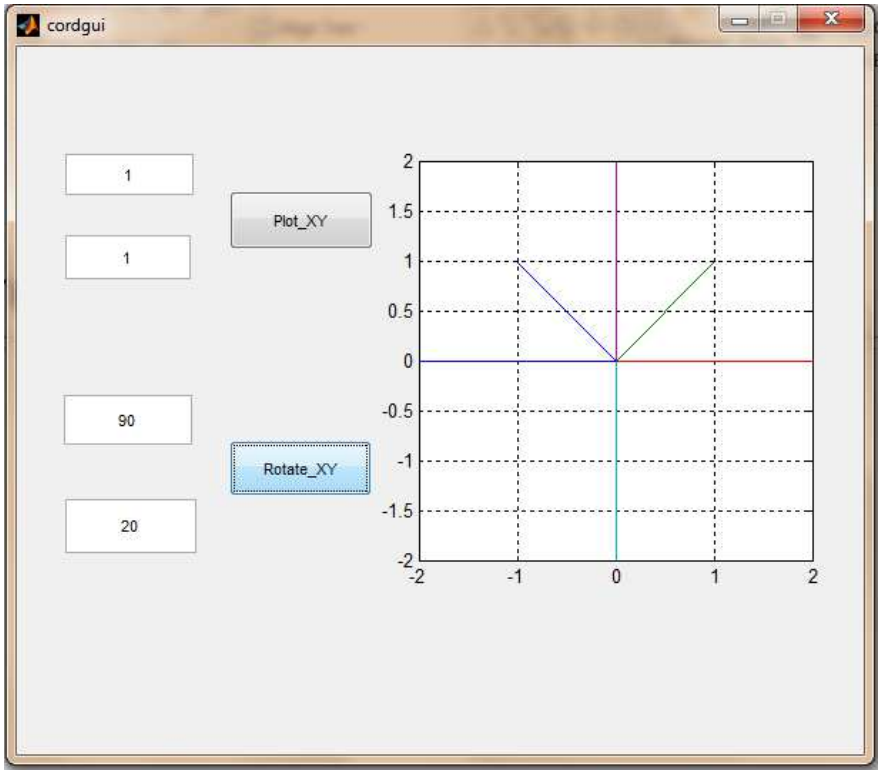

Figure 2.5: CORDIC output for rotaion mode.

The VHDL hardware implementation of CORDIC algorithm is reported in this section. In order to maintain sufficient accuracy of this design, we describe the output of cosine and sine function in 16-bit signed numbers.

2.3 VHDL Hardware for CORDIC Algorithm

The VHDL hardware implementation of CORDIC algorithm is reported in this section. In order to maintain sufficient accuracy of this design, we describe the output of cosine and sine function in 16-bit signed numbers.

2.3.1 Range for X and Y

According to the aim, requirement at the end of CORDIC hardware development, to calculate cosine value of an inputted angle. So the output will in the range of -1 to 1. To avoid the overflow in hardware, it has been extended the range from -2 to 2. This range is mapped with 16-bit of signed number to produce accurate output for further processing.

The outputs comes out in the range of 2 to +2. The results can be obtained as follows:

$2.0 \equiv 2^{15}$

$0.5 \equiv \frac{2^{15}}{2.0} \bullet 0.5 = 8192$

2.3.2 Range of Angle

Any angle cosine can be found, if the cosine of 180 to 180 degrees is calculated. So, the angle range has been set for -180 to 180 degree. As our system takes 16-bit signed input, we have to convert it into binary 16-bit signed number.

The input angle takes values from 180 degree to +180 degree where:

8000 (Hex) = 180 degree

$EFFF$ (Hex) = +180 degree

So, the calculation of equivalent 16-bit binary number as follows

$360\,\text{deg} \equiv 2^{16}$

$1\,\text{deg} \equiv \frac{2^{16}}{360}$

$90\,\text{deg} \equiv \frac{2^{16}}{360} \bullet 90 = 16384(dec) = 4000(hex)$

Now the output for $x_i = 0.5$, $y_i = 0.5$ and angle = 60 in VHDL is shown in the fig.

This can be better understood if it has a pictorial representation of vector rotation. This pictorial view has been developed by MATLAB simulation and described below.

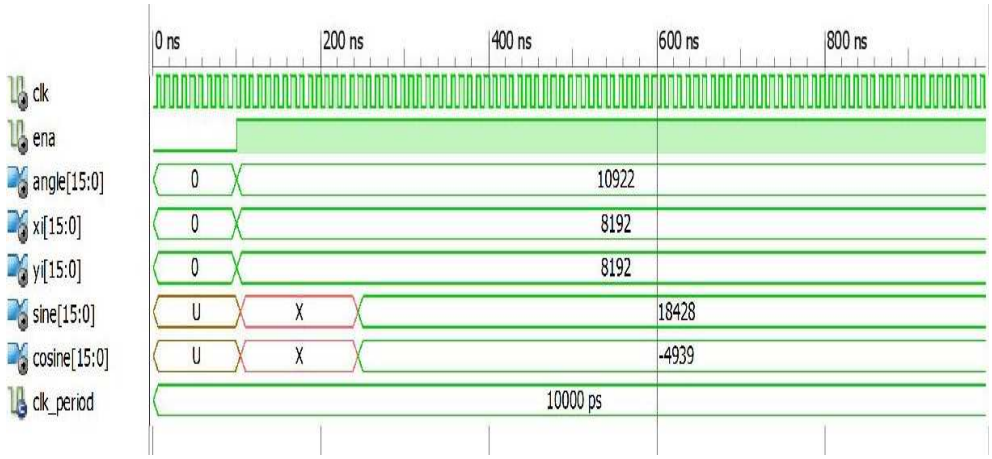

Figure 2.6: VHDL CORDIC core output.

2.4 MATLAB simulation for CORDIC Core

For better understanding of CORDIC core output, advantages of MATLAB is taken [14]. As it is possible to give input from MATLAB to ModelSim for simulation of VHDL core as shown in below fig [2.7].

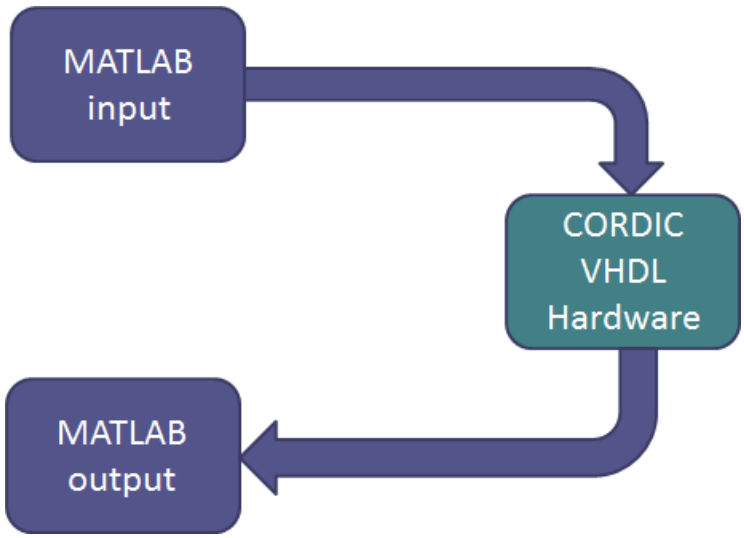

Figure 2.7: MATLAB simulation of VHDL core.

The input is taken through MATLAB, and that has been sent to VHDL hard-

ware and the output of VHDL hardware has been taken back to MATLAB and a vector plot is drowned as shown in fig [2.8][2.9].

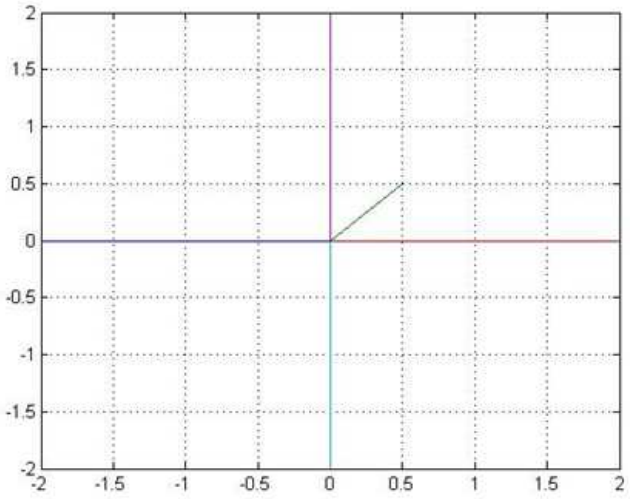

Figure 2.8: MATLAB input to VHDL core.

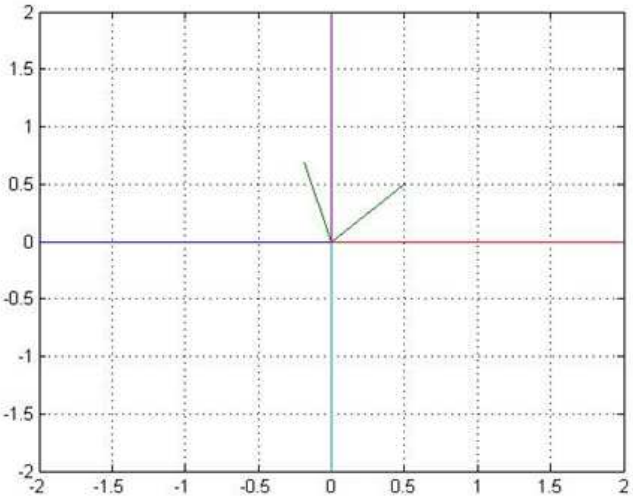

Figure 2.9: VHDL core output to MATLAB

From the MATLAB simulation we have found that this core is doing the same as MATLAB GUI developed for CORDIC algorithm. So having the final aim in focus we should create a hardware for cosine function from CORDIC algorithm. Thus, we designed the hardware for cosine function and described in next section.

2.5 Calculation of Cosine and Sine Function using CORDIC Algorithm

The sine and cosine function can be calculate in rotation mode [5]. The final output of CORDIC rotation mode operation can be summarized as follows.

$$\lim_{x\to\infty}\begin{pmatrix} x_n \\ y_n \\ w_n \end{pmatrix} = K \times \begin{pmatrix} x_0 cosw_0 - y_0 sinw_0 \\ y_0 sinw_0 + x_0 cosw_0 \\ 0 \end{pmatrix} \tag{2.13}$$

So, if we put some specific values in above equation sine and cosine function. The final output as sine and cosine function is calculated below. Where $x_0 = K$, $y_0 = 0$ and $x_0 = \theta$ is taken.

$$\lim_{x\to\infty}\begin{pmatrix} x_n \\ y_n \\ w_n \end{pmatrix} = \begin{pmatrix} cosw_0 \\ sinw_0 \\ 0 \end{pmatrix} \tag{2.14}$$

Calculation of cosine and sine function has been validated with MATLAB. From the above simulation, it concludes that using this algorithm a core can be developed to calculate cosine and sine function. In this work, we are considering only the cosine function so we developed core that will produce only cosine output for a particular input. So the cosine core is designed.

The input for cosine and sine function will be 16-bit each of angle, x_i and x_i. But to get sine and cosine value we have make constant x_i and y_i. This conversion is described below.

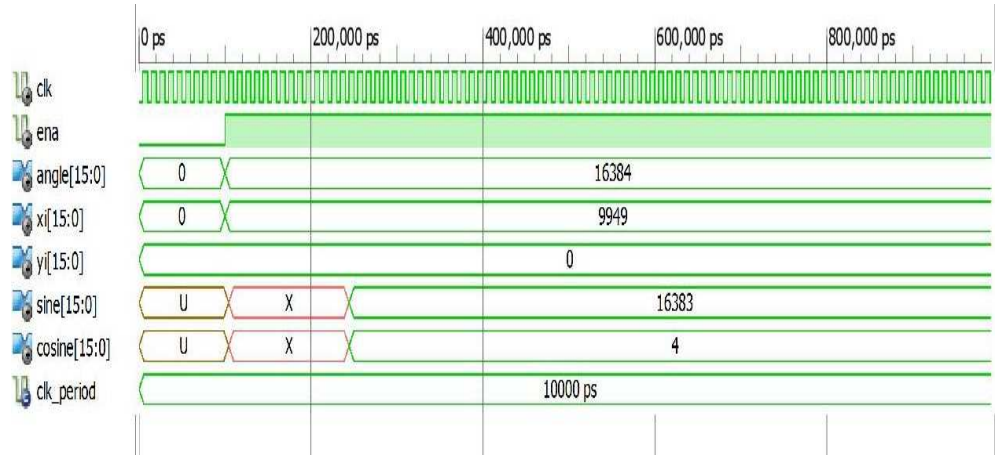

Figure 2.10: VHDL core for Sine and Cosine function.

From the conclusion of above equation the inputs will as follows

$x_i = \frac{1}{K} = \frac{1}{1.6467} \approx 0.60725$

$y_i = 0$

$w_i = 0$

the algorithm computes:

$[X_j, Y_j, Z_j] = [cos\theta, \sin\theta, 0]$

The input values for Z varies from 180degree to +180 degree where:

8000(Hex) = 180 degree

$EFFF$(Hex) = +180 degree

Co-ordinate values varies from 2 to +2 range. The K represented in this format yields the value as:

$x_i = 2^{15} \times K = 9949(dec) = 26DD(hex)$

- Example: Calculate the sine and cosine function of angle +90 degree. The value of angle in mapping format will be:

  $360\,\text{deg} \equiv 2^{16}$

  $1\,\text{deg} \equiv \frac{2^{16}}{360}$

  $90\,\text{deg} \equiv \frac{2^{16}}{360} \bullet 90 = 16384(dec) = 4000(hex)$

The CORDIC core will calculate the value of sine and cosine for the angle

$z_i = 4000$:

$Sin : 16383(dec) = 3FFF(hex)$

$Cos : 4(dec) = 0004(hex)$

The output of the core will in 2 to +2 range. The output can be derived into the original value as follow:

$2^{15} \equiv 2.0$,

$16383 \equiv \frac{2.0}{2^{15}} \bullet 16383 = 0.99993$

$4 \equiv \frac{2.0}{2^{15}} \bullet 4 = 0.00024$

But, the actual output is 1 and 0 respectively. Some common angles is tabulated in table[2.1].

Table 2.1: Sin/Cos outputs for some common angles

Angle	Sine (Hex)	Cosine (Hex)	Sine (Dec)	Cosine (Dec)
-180 (8000) deg	0002	BFFC	0.00012	1.00024
-150 (9555) deg	E002	C891	0.49987	- 0.86614
-135 (A000) deg	D2BF	D2C0	0.70709	0.70690
-120 (AAAA) deg	C891	E000	- 0.86614	- 0.50000
-90 (BFFF) deg	BFFC	FFFE	- 1.00024	0.00012
-60 (D555) deg	C891	1FFD	- 0.86614	0.49981
-45 (E000) deg	D2BE	2D41	- 0.70715	0.70709
-30 (EAAA) deg	E000	376F	- 0.50000	0.86614
0 (0000) deg	0004	3FFF	0.00024	0.99993
30 (1555) deg	1FFD	376F	0.49981	0.86614
45 (2000) deg	2D41	2D42	0.70709	0.70715
60 (2AAA) deg	376F	1FFD	0.86614	0.49981
90 (4000) deg	3FFF	0004	0.99993	0.00024
120 (5555) deg	376F	E002	0.86614	- 0.49981
135 (6000) deg	2D40	D2BF	0.70715	- 0.70709
150 (6AAA) deg	1FFF	C891	0.49981	- 0.86614
180 (7FFF) deg	0003	BFFF	0.00018	- 0.99993

2.6 Summary

In this chapter, CORDIC algorithm is described. CORDIC algorithm is validated using MATLAB, and a GUI is developed. Sine and cosine function calculated using CORDIC algorithm and hardware made for same. For better visualization of hardware output, MATLAB has been interfaced with VHDL core and validated perfectly. Some of the standard angles have been calculated and hence tabulated. This hardware is giving magnificent accuracy.

Chapter 3

DCT Computation

DCT represents a finite number of data sequence into sum of cosine functions strength oscillating at different discrete frequencies called as coefficients. DCT has many application in signal processing. One of the vital application of DCT is compression of image in JPEG. Due to the advantage of frequency domain representation, the higher frequencies can be removed to make compact representation which is possible in case of image representation because high frequency cannot be detectable by human eyes. The cosine function is utilized from sine and cosine function for compression, since things being what they are nearly less cosine functions are required to inexact a distinctive signal.

3.1 1D & 2D DCT

DCT is a linear transform, so it is an invertible function. There are different types 1D DCT available, and some of them are described below

3.1.1 DCT-I

$$X_k = \frac{1}{2}(x_0 + (-1)^k x(N-1)) + \sum_{n=1}^{N-2} x_n cos\left(\frac{\pi}{(N-1)} nk\right), \quad \text{where} \quad k = 0, 1, 2, ..., N-1. \tag{3.1}$$

DCT-I is equivalent to $(2N-2)$ DFT real numbers. DFT of the sequence is an even symmetry of DCT. If DCT of a sequence has 7 elements $abcdefg$ then, DFT sequence will be $abcdefgfedcb$. The DCT-I is defined for sequence less than 2. Multiplication of a factor $\sqrt{1/(N-1)}$ makes it orthogonal.

So, the boundary conditions for DCT-I:

x_n is even around $n=0$ and even around $n=N-1$,

X_k is even around $k=0$ and even around $k=N-1$.

3.1.2 DCT-II

$$X_k = \sum_{n=1}^{N-1} x_n cos\left(\frac{\pi k}{N}\left(n+\frac{1}{2}\right)\right), \quad \text{where} \quad k=0,1,2,...,N-1. \tag{3.2}$$

DCT-II also referred as The DCT due to its frequent use in signal processing. It is equivalent to $4N$ DFT. Multiplication of a factor $\sqrt{1/N}$ makes it orthogonal.

So, the boundary conditions for DCT-II:

x_n is even around $n=-1/2$ and even around $n=N-1/2$,

X_k is even around $k=0$ and odd around $k=N$.

3.1.3 DCT-III

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n cos\left(\frac{\pi n}{N}\left(k+\frac{1}{2}\right)\right), \quad \text{where} \quad k=0,1,2,...,N-1. \tag{3.3}$$

This is the inverse of DCT-II, so it also called as The inverse DCT. It multiplied by $\sqrt{1/N}$ makes it orthogonal.

So, the boundary conditions for DCT-III:

x_n is even around $n=0$ and odd around $n=N$,

X_k is even around $k=-1/2$ and even around $k=N-1/2$.

3.1.4 DCT-IV

$$X_k = \sum_{n=1}^{N-1} x_n cos\left(\frac{\pi}{N}\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}\right)\right), \quad \text{where} \quad k = 0, 1, 2, ..., N-1. \tag{3.4}$$

$\sqrt{1/N}$ can be multiply to make it orthogonal. DCT-IV has another form, where data sequence from different dictionary are overlapped, and called the modified discrete cosine transform (MDCT).

So, the boundary conditions for DCT-III:
x_n is even around $n = -1/2$ and odd around $n = N - 1/2$,
X_k is even around $k = -1/2$ and odd around $k = N - 1/2$.

3.1.5 DCT-V-VIII

DCTs of type's $I - IV$ has the point of symmetry uniformly on both sides. They are even or odd from data point. But, DCTs of types $V - VIII$ entire limits which are even or odd around a data point for one limit. They are centered on two data points for the other limit.

3.2 2D DCT using 1D DCT

DCT can be found in more than one dimension data sequence by expanding the 1D DCT equations. A one-dimensional data sequence like speech signal produces one dimension DCT coefficients. These coefficients are also invertible to data sequence as DCT is a unitary transformation.

Two-dimensional DCT can found in the two-dimensional data sequence. Image is one of the vital examples of two-dimensional signals. Two-dimensional DCT coefficients required for compression techniques in images because it is possible to discard different discrete coefficients from the DCT coefficient. So the DCT has great importance in signal processing technologies [10][11].

Two-dimensional DCT can be calculated from one-dimensional DCT coefficient. This technique has been described in section below

$$X(u,v) = \sum_{i=0}^{N-1} \left(\alpha(\xi) \left(\sum_{j=0}^{M-1} \alpha(\xi) y_n cos \left[\frac{\pi v}{N} \left(j + \frac{1}{2} \right) \right] \right) \left(x_n cos \left[\frac{\pi u}{M} \left(i + \frac{1}{2} \right) \right] \right) \right.$$

$$\text{where} \quad u = 0, 1, 2, ..., N-1; v = 0, 1, 2, ..., M-1.$$

$$\text{and} \quad , \alpha(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } \xi = 0 \\ 1 & \text{otherwise .} \end{cases} \tag{3.5}$$

In the above function if we calculate first row and column-wise, then, using $1D$ DCT coefficient we can calculate $2D$ DCT coefficient from a $2D$ given information matrix. So, the calculation is done first horizontally then vertically.

So, in case of image signal, the information matrix is $2D$. But from the literature review it has been seen that the 8×8 matrix gives better result in image processing technique. Thus, we conclude from the above discussion that our hardware structure will compute DCT of the information matrix of 8×8. Now, the primary aim of the work is to design $1D$ DCT hardware of eight data sequence.

3.3 Hardware for 1D DCT

One-dimensional DCT has been found using DCT-II. The DCT is determined by the cosine function. This cosine function is calculated using the CORDIC algorithm. The cosine core is repeatedly called by to compute the DCT.

In this work, the $1D$ DCT has been computed for a data sequence of length of eight because the image processing mostly uses 8×8 mask. This 8×8 has its advantages that have been supported by many researcher. Also, we are taking the MATLAB output as a reference that calculates $2D$ DCT of images in 8×8 frame.

Hardware takes a data sequence in 16-bit binary signed number format and produces the output coefficients in 32-bit binary signed number.

Figure 3.1: VHDL core output for 1D DCT.

The calculation for actual values of $1D$ DCT coefficients is determined on the basis of signed number. The number now must be

$$2^{15} \times 2(\text{output of VHDL core}) = 2.0$$

$$=> X_k = \frac{2}{2^{16}} \times \text{VHDL output}$$

Now, From the VHDL output

$$X_0 = \frac{2}{2^{16}} \times 12408606$$

$$=> X_0 = 378.6557.$$

We have considered MATLAB as reference for the calculation of DCT coefficient. The comparison of MATLAB and VHDL output is done in table 3.1. MSE

Table 3.1: 1D DCT comparison between MATLAB inbuilt and VHDL core

Input	MATLAB Output	VHDL Output
172	378.6557	378.6806
94	23.9139	23.9013
202	26.9600	27.9893
95	-27.9863	-27.9893
163	-8.5388	-8.5288
10	84.9818	84.9514
242	-55.9411	-55.9621
95	161.9860	161.9160

of the DCT is calculated by MATLAB inbuilt function and VHDL hardware is 0.0042. This value is very negligible compared with the hardware system of 16-bit input.

3.4 Hardware for 2D DCT

The 2D DCT is calculated in this section with the assistance of 1D DCT coefficient. 1D DCT of a 2D data matrix is calculated, the 2D matrix is divided into eight 1D matrix row-wise, each 1D matrix contains eight input sequence. Then, from 1D array 1D DCT coefficients computed. Then, from the 1D DCT coefficients again 1D DCT coefficient calculated in column wise. The final output gives 2D DCT coefficient.

The result produce from the hardware has been examined in below. The table 3.2, 3.3, 3.4 represents the data matrix, MATLAB 2D DCT output and VHDL hardware output.

Therefore, from the above result, we can see that the DCT estimation of MATLAB and VHDL hardware output is almost same.

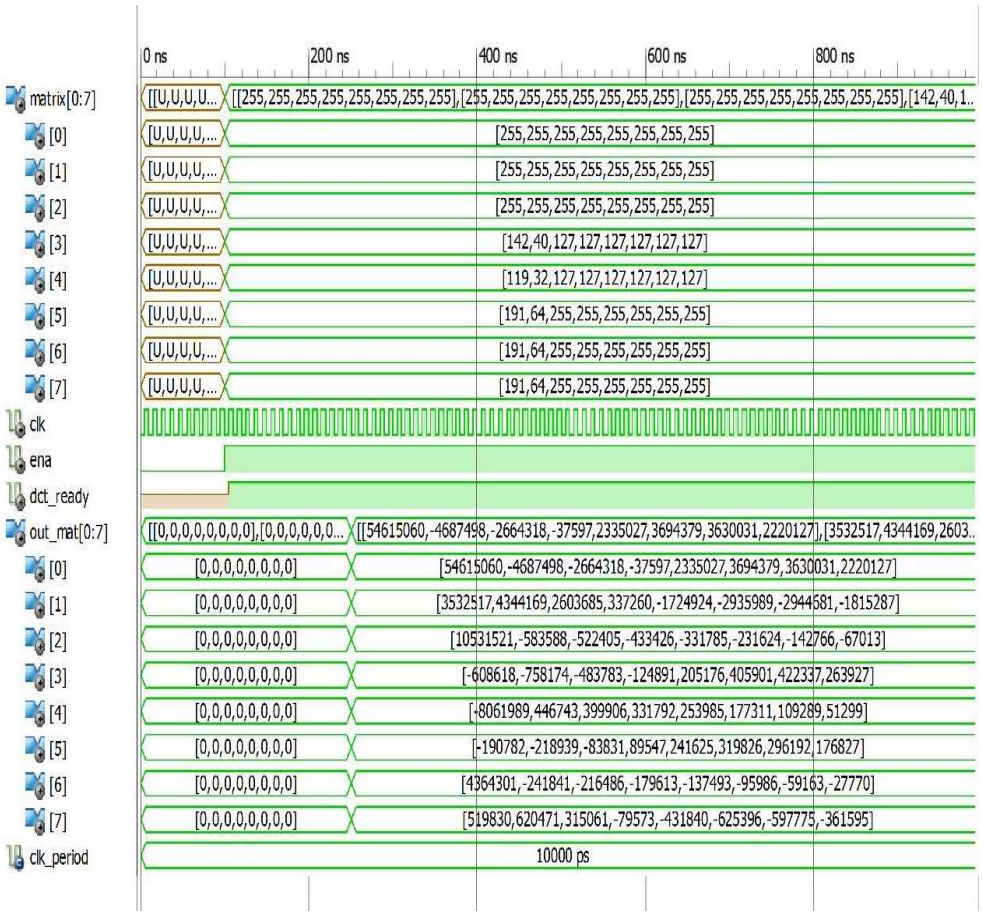

Figure 3.2: VHDL core ouput for 2D DCT.

3.5 Summary

In this chapter 1D and 2D DCT is described. Hardware of 1D and 2D DCT is designed with VHDL and the compared with inbuilt MATLAB function output. It observed that the output of VHDL and MATLAB inbuilt function is nearly same.

Table 3.2: 2D data matrix in spatial domain

255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
142	40	127	127	127	127	127	127
119	32	127	127	127	127	127	127
191	64	255	255	255	255	255	255
191	64	255	255	255	255	255	255
191	64	255	255	255	255	255	255

Table 3.3: 2D out matrix in spatial domain from MATLAB

1666.500	-143.0485	-81.28965	-1.154119	71.25000	112.7326	110.7917	67.76978
107.8060	132.5908	79.45464	10.29936	-52.64187	-89.60273	-89.89035	-55.42168
321.4144	-17.81216	-15.94606	-13.22961	-10.12586	-7.071932	-4.355077	-2.044379
-18.56972	-23.13547	-14.76098	-3.812855	6.258956	12.38293	12.88884	8.055617
-246	13.63283	12.20459	10.12550	7.749999	5.412622	3.333232	1.564700
-5.817057	-6.676044	-2.553418	2.733065	7.371214	9.756875	9.036205	5.395374
133.1342	-7.378039	-6.605077	-5.479886	-4.194269	-2.929290	-1.803932	-0.846809
15.85652	18.92844	9.607768	-2.428118	-13.17472	-19.08020	-18.23988	-11.03489

Table 3.4: 2D out matrix in frequency domain from VHDL

1666.719	-143.0510	-81.30853	-1.147369	71.25936	112.7434	110.7797	67.75289
107.8038	132.5735	79.45816	10.29235	-52.64050	-89.59927	-89.86453	-55.39816
321.3965	-17.80969	-15.94253	-13.22711	-10.12527	-7.068603	-4.356872	-2.045074
-18.57354	-23.13763	-14.76388	-3.811370	6.261474	12.38711	12.88870	8.054412
-246.0323	13.63351	12.20416	10.12548	7.751007	5.411102	3.333232	1.564700
-5.822204	-6.681488	-2.558319	2.732757	7.373809	9.760314	9.039062	5.565521
133.1878	-7.380401	-6.606628	-5.481353	-4.195953	-2.929260	-1.805511	-0.847473
15.86395	18.93527	9.614898	-2.428375	-13.17871	-19.08557	-18.24264	-11.03500

Chapter 4

4. SSIM Computation

The distortion measure in the image is very vital in the area of image processing. The main distortion or noise can produce or introduced by processing of the image. So, it is important to have the knowledge about the processed image before present it to the viewer. This measurement of image distortion mainly depends on the availability of the original image. In accordance with the availability of the reference image, the distortion measure can be of three types. These are either full-reference estimation, partial-reference or blind reference of noise.

In full-reference quality estimation, the original image is in the hand to compare with the noisy image. But, in case of partial-reference image quality estimation, some of the information is available to compare with a distorted image, whereas, in the blind-reference assessment, there is no information about the original image. Among these three types of assessment of image quality technique, full-reference image quality assessment has more applications in the area of image processing.

There are many types of algorithms are available for the full-reference image quality assessment. Mostly MSE, PSNR and SSIM have been used so far in the area of image processing. But, then some advantages have been found out in SSIM, which has been proposed by [1]. Due to the advantages of SSIM in the image, it has been adopted in this work.

4.1 SSIM in Spatial Domain

Image of an object is a function of its illumination and reflectance. But, the illumination does not depend on the structure of the object. To evaluate the SSIM of an image we have to separate out the image illumination factor. This illumination factor can be separate out by representing the image with average luminance and contrast.

Suppose we have two images x, and you with positive values. The SSIM measures the index between two images with three factors, and these are mean, variance and structure. The total SSIM of the image is given by multiplication these three factors [1].

$$S(x,y) = f\big(l(x,y), c(x,y), s(x,y)\big) \tag{4.1}$$

Hare, x = original image and y = distorted image.
$S(x,y) = \text{SSIM}, l(x,y) = \text{luminance}, c(x,y) = \text{contrast}, s(x,y)\text{contrast}.$

The luminance of an image is defined in terms of its mean intensity level.

$$\mu = \frac{1}{N}\sum_{i=0}^{N-1} x_i \tag{4.2}$$

Now, the luminance comparison between two images is as follows

$$l(x,y) = \frac{2\mu_x\mu_y + c1}{\mu_x^2 + \mu_y^2 + c1} \tag{4.3}$$

Luminance effect can be reduced by subtracting it from each pixel value that will represent standard deviation. Let the standard deviations are σ_x and σ_y. Then, the expression will be as follows:

$$c(x,y) = \frac{2\sigma_x\sigma_y + c2}{\sigma_x^2 + \sigma_y^2 + c2} \tag{4.4}$$

So, final comparison has to do after normalizing the distorted and original image from its standard deviation. Which can be written as $(x - \mu_x)/\sigma_x$. The final normalized expression can be written as

$$s(x,y) = \frac{\sigma_{xy} + c3}{\sigma_x \sigma_y + c3} \tag{4.5}$$

Now, the final expression of SSIM can be written as

$$S(x,y) = l(x,y)^{\alpha} \times c(x,y)^{\beta} \times s(x,y)^{\gamma} \tag{4.6}$$

The values of α, β and γ are taken to adjust according to the importance of the parameter. But here the values for α, β and γ are taken as unity. Then, the final spatial domain SSIM will be

$$S(x,y) = \frac{2\mu_x\mu_y + c1}{\mu_x^2 + \mu_y^2 + c1} \times \frac{2\sigma_{xy} + c2}{\sigma_x^2 + \sigma_y^2 + c2} \tag{4.7}$$

The expression for other parameters is written below.

$$\begin{aligned} c1 &= (K1 \times L)^2, \\ c2 &= (K2 \times L)^2, \\ c3 &= c2/2 \\ L &= 255 \\ K1 &= 0.01 \\ K2 &= 0.03. \end{aligned}$$

4.2 SSIM from DCT co-efficient

It is preferable to calculate the SSIM in a representation domain like DFT, DCT, and wavelet. As our concern about the DCT domain in this work, so we will estimate SSIM in this domain. From the above section, we have seen that SSIM is the combination of mean, standard deviation and co-variance. It would be easy to calculate SSIM in DCT domain if we able to estimate these parameters from DCT coefficient.

DCT of a data sequence $x \epsilon R$ is written below

$$X_k = \sum_{n=0}^{N-1} \alpha(\xi) y_n cos\left[\frac{\pi v}{N}\left(n+\frac{1}{2}\right)\right]$$
$$\text{where,} \quad \alpha(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } \xi = 0 \\ 1 & \text{otherwise .} \end{cases} \tag{4.8}$$

DCT is a unitary transform. It obeys Parseval theorem. Thus, from the equation [4.8] and the property of DCT we can draw the relation between the spatial parameter of SSIM and DCT coefficients[15]. The parameters are written in terms of DCT coefficient in the section below.

$$\mu = \frac{\sum_{i=0}^{N-1} x_i}{n} = \frac{X(0)}{\sqrt{N}}$$
$$\sigma_x^2 = \frac{\sum_{k=1}^{N-1} X_k^2}{N-1} \tag{4.9}$$
$$\sigma_{xy} = \frac{\sum_{k=1}^{N-1} X_k Y_k}{N-1}$$

So, using these parameters we can calculate SSIM in DCT domain.

4.3 Hardware for parameter calculation of SSIM

Parameters of SSIM are mean, standard deviation and cross-covariance. From the equation [4.9] parameters is calculated in the DCT domain. This hardware takes inputs, two 8×8 matrix in the spatial domain and converts into DCT domain and then calculates the parameters in 32-bits of binary numbers. The conversion of a binary number to decimal number is described below.

Decimal Output = binary Output/2^{15}

Inputs of the hardware are taken same matrix to check co-variance of hardware with standard deviation, as we know that co-variance of same matrix is same as its

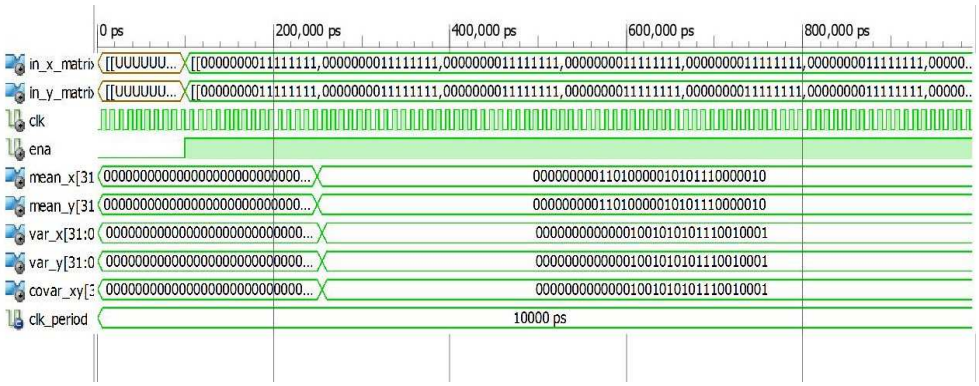

Figure 4.1: VHDL core ouput for SSIM parameter calculation.

Table 4.1: Parameter comparison between MATLAB inbuilt function and VHDL core

parameters	MATLAB output	VHDL output
mean	208.3125	208.3399
variance	9.3217	9.3403
co-variance	9.3217	9.3403

standard deviation. Now, we can observe from the table 4.1 that its co-variance is same as standard deviation.

4.4 Hardware for SSIM

VHDL Hardware of SSIM in DCT domain designed from the parameter calculation hardware. Here the inputs are two matrix of 8×8 element, and the output is 64 bit binary number. The calculation of is described below.

Decimal SSIM Output = binary SSIM Output/2^{15}

The above tabulation shows that the value of inbuilt MATLAB SSIM and VHDL hardware output nearly same.

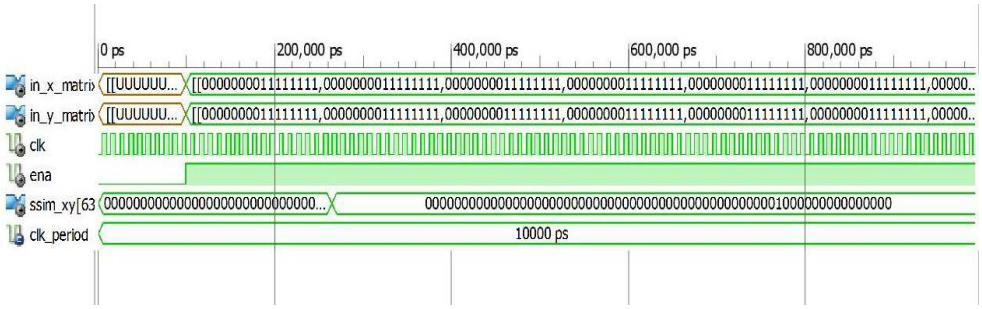

Figure 4.2: VHDL core output for SSIM calculation.

Table 4.2: SSIM comparison between MATLAB in built function an VHDL core

Sl.no	MATLAB SSIM value	VHDL SSIM value
1	1	1
2	0.9528	0.9638
3	0.9628	0.9706
4	0.9576	0.9665
5	0.5936	0.6084
6	0.2062	0.1927
7	0.0777	0.0523
8	0.0276	0.0198
9	0.9562	0.9632
10	0.2372	0.2216

4.5 Summary

In this chapter basics of SSIM are described. The compressed statistical parameter in terms of DCT coefficient is defined and designed in VHDL. SSIM hardware also designed for the statistical parameter hardware. Comparison is done between MATLAB inbuilt function and VHDL hardware output, and it has been found out these nearly same.

Chapter 5

Test bench for Evaluation of SSIM Computation through Hardware

A test bench is a platform in which the system under development is examined with the assistance of software and hardware tools. It is also used to check the correctness of hardware or software. It has the advantage mimic the real hardware, which can be used to realize and optimize a hardware system. In this work, VHDL test bench is used to simulate hardware for estimation of similarity of the image. The test bench of hardware is taking data from MATLAB generated.DAT file. The output generated from the VHDL core is saved in a .TEXT file. Then, the output is realized in MATLAB environment.

5.1 Accuracy for Spatial & Frequency domain measurement

Hardware SSIM value is compared with MATLAB inbuilt SSIM function and a plot is drawn. Average of error is calculated as 0.00128 and shown in figure below.

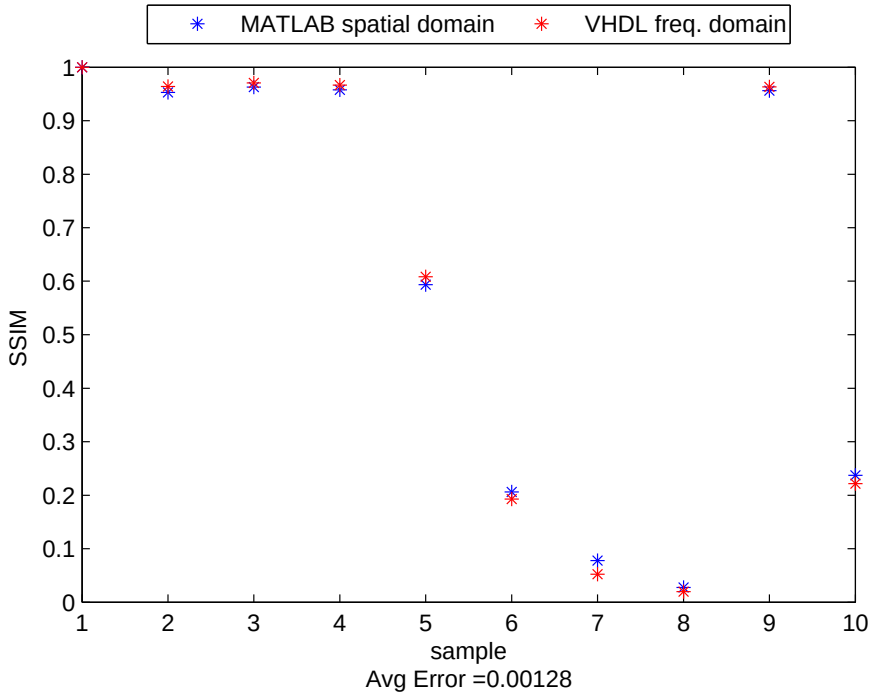

Figure 5.1: Comparision of SSIM computation through frequency domain approach in FPGA and spatial approach in MATLAB.

5.2 Time Complexity between MATLAB inbuilt function and FPGA Hardware

Time complexity between SSIM computation in MATLAB R2012b and FPGA design is compared for ten samples and shown in below fig [5.2]. The average time improved in percent by FPGA design is 26.0925%.

5.3 Summary

Hardware is validated using test bench. The accuracy of the designed hardware is shown, and it found significantly less. The error calculation and elapsed time plot are drawn.

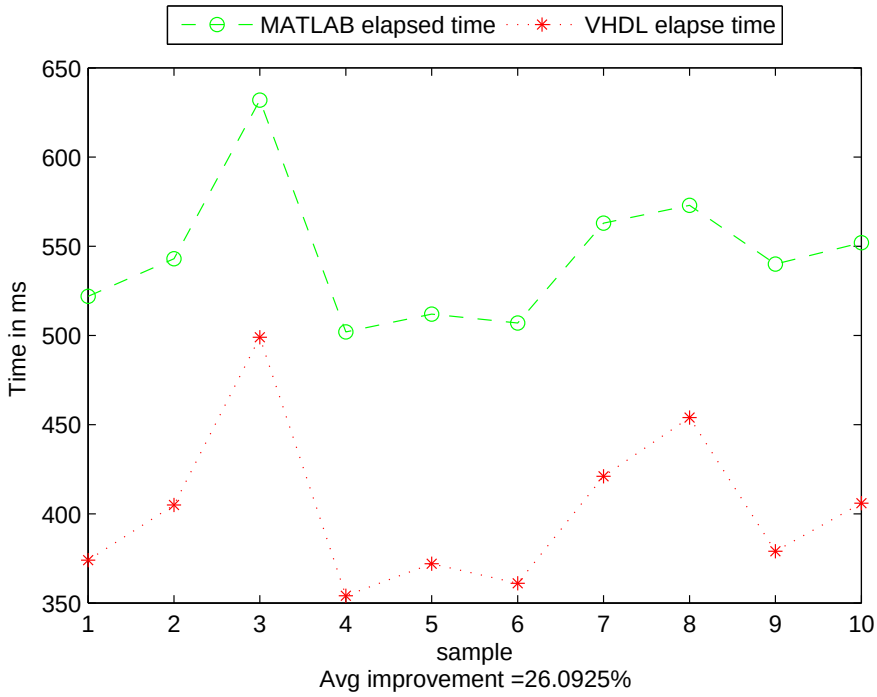

Figure 5.2: Comparision of time elapsed betwen MATLAB 2012b and FPGA Design.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Hardware for CORDIC algorithm, 1D DCT, 2D DCT and SSIM is designed. Results found from the hardware simulation are significantly same as MATLAB inbuilt results. For better visualization of hardware output from the ModelSim simulator was processed in MATLAB for CORDIC algorithm. But, in case of SSIM hardware simulation in ModelSim was not possible due to its lack of capacity in ModelSim PE Student Edition 10.4a. It can be developed by using a full version of ModelSim. Finally, the plot has been drawn for error estimation of hardware output comparisons and time complexity with the MATLAB inbuilt SSIM function.

6.2 Future Work

This work concentrated on design consideration and accuracy. Time complexity may again decreased by using some efficient algorithm. This model can be implemented on FPGA board. SSIM also can be calculated in wavelet and curvelet dictionary in hardware. This hardware can be implemented in a system on the chip for image processing. The design of simulator for the analysis of SSIM for the whole image.

Bibliography

[1] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. "Image quality assessment: from error visibility to structural similarity", *Image Processing, IEEE Transactions*, 13.4 (2004), 600-612.

[2] Alain Hore, and Ziou Djemel. "Image Quality Metrics: PSNR vs. SSIM", *ICPR.* 34, (2010).

[3] Crookes, D., et al. "Design and implementation of a high level programming environment for FPGA-based image processing", *IEE Proceedings-Vision, Image and Signal Processing*, 147.4 (2000): 377-384.

[4] Pramod K. Meher, Javier Valls, Tso-Bing Juang, K. Sridharan, and Koushik Maharatna. "50 years of CORDIC: Algorithms, architectures, and applications", *Circuits and Systems I: Regular Papers, IEEE Transactions*, 56.9 (2009): 1893-1907.

[5] Jean-Michel Muller."Elementary functions: algorithms and implementation", *Springer Science & Business Media*, (2006).

[6] Jack E.Volder. "The CORDIC trigonometric computing technique", *Electronic Computers, IRE Transactions*, 3 (1959): 330-334.

[7] Fayez Gebali. "Algorithms and Parallel Computing", *John Wiley & Sons*, 84. (2011).

[8] Zhou Wang, and Alan C. Bovik. "A universal image quality index", *Signal Processing Letters, IEEE*, 9.3 (2002): 81-84.

[9] Mathai Joseph, and P. Pandya. "Finding response times in a real-time system", *The Computer Journal*, 29.5 (1986): 390-395.

[10] Syed Ali Khayam. "The discrete cosine transform (DCT): theory and application", *Michigan State University*, (2003).

[11] Peng Chungan, Cao Xixin, Yu Dunshan, and Zhang Xing. "A 250MHz optimized distributed architecture of 2D 8x8 DCT", *ASIC, 2007. ASICON'07. 7th International Conference*, (2007).

[12] Volnei A Pedroni. "Digital electronics and design with VHDL", *Morgan Kaufmann*, (2008).

[13] Donald G. Bailey. "Design for embedded image processing on FPGAs", *John Wiley & Sons*, (2011).

[14] Sreejith M. "Development of an FPGA based Image Processing Intellectual Property Core", *NIT, Roukela*, (2014).

[15] Sankarasrinivasan S. "Compressed Measurement of Structural Similarity Index", *NIT, Roukela*, (2014).

[16] Phil Lapsley, Jeff Bier, Edward A. Lee, and Amit Shoham. "DSP processor fundamentals", *IEEE SPECTRUM*, (1998).